An introduction to interactive programming in Python

第一周一个quiz，之后每周两个quiz

可以先做Practice exercises

codeskupltor：编程工具

vizmode：可视化的debug工具

string：用””包住，或者用’’

如果string中间出现’，可以在’之间加上\来区分

e.g. ‘There\’s a snake in my boot!’

%s替代字符串内字符段：

e.g.

string\_1 = "Camelot"

string\_2 = "place"

print "Let's not go to %s. 'Tis a silly %s." % (string\_1, string\_2)

%f替代函数变量：

def tax(bill):

"""Adds 8% tax to a restaurant bill."""

bill \*= 1.08

print "With tax: %f" % bill

return bill

codeskulptor:

保存文件的方法——点击左数第二个，disk按钮

1. 把上方的URL复制粘贴出来或者添加书签

2. 点击左数第三个，下载到本地

取回程序：

1. 把保存的URL输入进网址

2. 左数五个从本地浏览文件

对已同一段代码，反复保存后，URL最后version数字会增加

如果需要建立一个全新的URL，点击左数第四个suitcase button

点击行号可以折叠一段代码

Vizmode：

Codeskulptor右上角最右边按钮

左上角最右的扳手按钮：debug——把控制台分成两部分，下面为可视化窗口

运行后，右上控制台出现begin和end，点击begin，左边对应到了第一行

然后点击左上角的右箭头，可以向下执行一个命令

frame：变量及其定义之间的映射

左上角最右侧的箭头：在breakpoint/event之间跳跃

直接点击左侧的行标号，行标号变红，设置了一个breakpoint

debug结束，点击扳手回到正常模式

python内置函数

可以在codeckulpter右上角doc里面查看math module和random module，有各类数学内置函数和生成随机数函数的信息

弹出输入框：

a = raw\_input(“Where do you want to go?”)

(-1) \*\* True = -1

(-1) \*\* False = 1

from math import sqrt

之后可以直接print sqrt(25)之类的，不需要print math.sqrt(25)

from math import \*

就Import了math的所有子函数，之后取用都不用math.了

用global variable并不是一个好主意

beyond CodeSkulptor

https://www.python.org/

下载python，打开会有一个shell界面，可以在这个界面直接输入指令并执行，也可以运行别的Module文件，结果显示在shell

useful modules:

https://wiki.python.org/moin/UsefulModules#GUI

game libraries:

https://wiki.python.org/moin/PythonGameLibraries

recommend: TkInter, WxPython

PyGame: 最广泛使用的游戏库

(1) Arithmetic expressions

print 3, -1, 3.14159, -2.8 3 -1 3.14158 -2.8 数字直接没有逗号

两类数字：

①integer 整型 int()

②decimal 浮点型 float()

检测数字类型：

print type(3), type (3.0) <type ‘int’> <type ‘float’>

把浮点型转换成整型（取出整数部分）

print int(3.14159), int(-2.8) 3 -2

把整型转换成浮点型（加上.0）

print float(3), float(-1) 3.0 -1.0

把数字转换成字符串

e.g. 把时间转换成0X:00形式：

hour = 3

ones = hour % 10

tens = hour // 10

print str(tens),str(ones),”:00” 0 3 :00

print str(tens) + str(ones) + “:00” 03:00

python只用15位小数的精度，超出部分都会被抛弃——存在近似误差

算术运算符：

加 +

减 -

乘 \*

除 /

指数 \*\*

整除 //

模运算（余数） %

print 1+2, 3-4, 5\*6, 2\*\*5 3 -1 30 32

除法：

python 2中，如果其中一个运算的数是浮点数，结果就是浮点数

print 1.0/3, 5.0/2.0 0.333333333333333 2.5

如果两个数都是整型，那么结果是整型（精确值rounded down的结果）

print 1/3, 5/2, -7/3 0 2 -3

In Python 2.7: 27 / 10   and 27 // 10 both give the same result of 2  
In Python 3, these would return 2.833333333 and 2 respectively

算术表达式：数字/ 数字+运算符

print 1+2\*3, 4.0-5.0/ 6.0, 7\*8+9\*10 7 1.666666666666665 146

优先级： (), \*\*, \*, /, +, - please excuse my dear aunt sallie 首字母

模运算：

优先级高于加减

对负数进行模运算，结果在0和被除数之间：  
e.g. print -3 % 7 结果是4

(2) Variables

用于储存值、帮助理解某个值的意义

有效命名：字母，数字，下划线，必须以字母或者下划线开头，区分大小写（通常用小写）

一般用两个单词之间加下划线来帮助理解

赋值：variable=value（右边可以用数字表达式）

修改赋值： variable = variable +1之类的 也可以写作 variable += 1

其他运算同理。在现值基础上对自己运算并更新

ps：判断是否相等：==双等号（比较逻辑值）

http://www.codeskulptor.org/#examples-variables.py

# variables - placeholders for important values

# used to avoid recomputing values and to

# give values names that help reader understand code

# valid variable names - consists of letters, numbers, underscore (\_)

# starts with letter or underscore

# case sensitive (capitalization matters)

# legal names - ninja, Ninja, n\_i\_n\_j\_a

# illegal names - 1337, 1337ninja

# Python convention - multiple words joined by \_

# legal names - elite\_ninja, leet\_ninja, ninja\_1337

# illegal name 1337\_ninja

# Temperature examples

# convert from Fahrenheit to Celsuis

# c = 5 / 9 \* (f - 32)

# use explanatory names

temp\_Fahrenheit = 212

temp\_Celsius = 5.0 / 9.0 \* (temp\_Fahrenheit - 32)红色部分如果5/9，会得出0

print temp\_Celsius 100.0

# test it! 32 Fahrenheit is 0 Celsius, 212 Fahrenheit is 100 Celsius

# convert from Celsius to Fahrenheit

# f = 9 / 5 \* c + 32

temp\_Celsius = 100

temp\_Fahrenheit = 9.0 / 5.0 \* temp\_Celsius + 32 212.0

print temp\_Fahrenheit

# test it!

(3) Function

定义一段代码作为一个function，然后反复调用

http://www.codeskulptor.org/#examples-functions.py

函数名尽量定义的容易理解

documentation string文档字符串：一般def语句之后，”””分行，表明意图

注释：表明怎么算的之类

e.g. 定义函数

def triangle\_area (base, height): 函数头（以def开头）

“””

this returns area of a triangle, given the lengths of base and height

“””

# triangle area formula

area= (1.0/2) \* base \* height 函数体（缩进量要相等）

return area

def+函数名(命名规则同变量名)+(参数)+:(意味着后面是一个新的代码块)

return + 变量：输出某个变量的值

* print statements can appear anywhere in your program and print a specified value(s) in the console. Note that execution of your Python program continues onward to the following statement. Remember that executing a print statement inside a function definition does **not** return a value from the function.
* return statements appear inside functions. The value associated with the return statement is substituted for the expression that called the function. Note that executing a return statement terminates execution of the function definition **immediately**. Any statements in the function definition following the return statement are ignored. Execution of your Python code resumes with the execution of the statement after the function call.

e.g. 调用函数

a1 = triangle\_area(3,8)

print a1

e.g.

def hello ():

print “Hello, world!” 没有输入参数，也没有输出值

hello() 输出”Hello world!”

h=hello()

print h 输出None

没有Return的语句，会返回None

load a module导入内置模块: import math或者import random之类的

random module:

exercise 1

def miles\_to\_feet(miles):

feet= miles \* 5280

print str(miles) + " miles equals "+str(feet)+" feet."

miles\_to\_feet(13)

miles\_to\_feet(57)

miles\_to\_feet(82.67)

13 miles equals 68640 feet.

57 miles equals 300960 feet.

82.67 miles equals 436497.6 feet.

直接输入函数，就可以执行了（这个函数没有return，因为最后只要print一句话）

（标准答案：

def miles\_to\_feet(miles):

"""Returns the number of feet in the given number of miles."""

return miles \* 5280

def test(miles):

"""Tests the miles\_to\_feet function."""

print str(miles) + " miles equals",

print str(miles\_to\_feet(miles)) + " feet."

test(13)

test(57)

test(82.67)

写两个函数，先return miles\*5280作为第一个函数的值，第二个函数再调用）

exercise 5

def circle\_circumference(radius):

import math

circumference=2\*math.pi\*radius

print "A circle with a radius of " + str(radius),

print "inches has a circumference of",

print str(circumference) + " inches."

circle\_circumference(8)

circle\_circumference(3)

circle\_circumference(12.9)

A circle with a radius of 8 inches has a circumference of 50.2654824574 inches.

A circle with a radius of 3 inches has a circumference of 18.8495559215 inches.

A circle with a radius of 12.9 inches has a circumference of 81.0530904626 inches

别忘记import math

exercise 7

def future\_value(present\_value, annual\_rate, years):

"""Tests the future\_value function."""

future\_value= present\_value \* (1.0+annual\_rate/100.0)\*\*years

print "The future value of $" + str(present\_value) + " in " + str(years),

print "years at an annual rate of " + str(annual\_rate) + "% is",

print "$" + str(future\_value) + "."

future\_value(1000, 7, 10)

future\_value(200, 4, 5)

future\_value(1000, 3, 20)

The future value of $1000 in 10 years at an annual rate of 7% is $1967.15135729.

The future value of $200 in 5 years at an annual rate of 4% is $243.33058048.

The future value of $1000 in 20 years at an annual rate of 3% is $1806.11123467.

注意数字类型（加.0）

exercise 9

def name\_and\_age(name,age):

return name+" is "+str(age)+" years old."

def test(name, age):

print name\_and\_age(name, age)

test("Joe Warren", 52)

test("Scott Rixner", 40)

test("John Greiner", 46)

Joe Warren is 52 years old.

Scott Rixner is 40 years old.

John Greiner is 46 years old.

(4) Logic

Boolean Logic 布尔逻辑：

只有TRUE/FALSE

逻辑运算符：NOT/AND/OR

e.g.

a=True

b=False

c=True

d=False

print not a False

print a and b False

print a or b True

print (a and b) or (c and (not d) True

比较运算符：可以比较数值、字符串等各种变量类型

>

<

>=

<=

== equal（单等号是赋值，双等号是比较运算符）

!= not equal to

e.g.

a=7>3

print a True

x=8

y=5

b=x>=y

print b True

c=”Hello” == ‘Hello’

print c True（比较的是字符串，和单引号双引号无关）

优先级： not > and > or

e.g. a or b and c等价于a or (b and c)等价于 b and c or a

(5) Conditionals 条件语句

e.g.

def greet(friend, money):

if friend and (money > 20):

print ”Hi!”

money=money-20

elif friend:

print “Hello”

else:

print “Ha ha”

money = money + 10

return money

money = 15

money = greet(True, money) Hello

print “Money:” , money 15（执行elif，没有多余20元）

money = greet(False, money) Ha ha

print “Money:”, money 25（执行else，增加10元）

money = greet (True, money) Hi

print “Money:” , money 5（执行if）

if 后面部分是true（后面的表达式或者布朗变量为true），执行冒号后面部分

如果False，检查elif是否true，否则执行 else后面部分

elif 可以有很多个，但是if和else只能有一个

e.g.

def is\_leap\_year(year):

if (year % 400) == 0:

return True

elif (year % 100) == 0:

return False

elif (year % 4) == 0:

return True

else:

return False

year = 2100

leap\_year = is\_leap\_year(year)

if leap\_year:

print year, “is a leap year”

else:

print year, “is not a leap year”

结果: 2100 is not a leap year

带逻辑判断的条件语句的简化：

e.g.

def g(a,b):

if a == True and b == True:

return False

else:

return True

等价于

def g(a,b):

if a and b:

return False

else:

return True

等价于

def g(a,b):

return not (a and b)

exercise1 偶数测试

def is\_even(number):

remainder=number % 2

return remainder==0

def test(number):

"""Tests the is\_even function."""

if is\_even(number):

print number, "is even."

else:

print number, "is odd."

exercise 3 whether lunchtime

def is\_lunchtime(hour,is\_am):

if hour==11 and is\_am==True:

return True

elif hour==12 and is\_am==False:

return True

else:

return False

def test(hour, is\_am):

"""Tests the is\_lunchtime function."""

print hour,

if is\_am:

print "AM",

else:

print "PM",

if is\_lunchtime(hour, is\_am):

print "is lunchtime."

else:

print "is not lunchtime."

test(11, True)

test(12, True)

test(11, False)

test(12, False)

test(10, False)

11 AM is lunchtime.

12 AM is not lunchtime.

11 PM is not lunchtime.

12 PM is lunchtime.

10 PM is not lunchtime.

函数可以直接return True/False（注意首字母一定大写），if后面比较语句的等号要用双等号。

exercise 5 whether intersect (assume a<=b, c<=d)

def interval\_intersect(a, b, c, d):

"""Returns whether the intervals [a,b] and [c,d] intersect."""

return (c <= b) and (a <= d)

def test(a, b, c, d):

"""Tests the interval\_intersect function."""

print "Intervals [" + str(a) + ", " + str(b) + "] and [" + str(c) + ", " + str(d) + "]",

if interval\_intersect(a, b, c, d):

print "intersect."

else:

print "do not intersect."

test(0, 1, 1, 2)

test(1, 2, 0, 1)

test(0, 1, 2, 3)

test(2, 3, 0, 1)

test(0, 3, 1, 2)

Intervals [0, 1] and [1, 2] intersect.

Intervals [1, 2] and [0, 1] intersect.

Intervals [0, 1] and [2, 3] do not intersect.

Intervals [2, 3] and [0, 1] do not intersect.

Intervals [0, 3] and [1, 2] intersect.

巧用and语句

(6) error treatment and test

如果遇到错误，错误行会标红

NameError可能由于：

拼错变量名、函数名，或者使用内置函数的时候没有先import内置模块

AttributeError可能由于：

内置模块内的函数名拼错

TypeError可能由于：

应该为数值的变量变成了字符串；函数参数不足

SyntaxError可能由于：

语法错误，例如函数def语句后面忘了冒号，=和==混用，变量名定义不符合规则，缺少缩进等等

(7) event-driven programming

start→initialize→wait→event occur→handler→quit→end

Events:

①input

e.g. button/ textbox

②keyboard

e.g. key down/up

③mouse

click/drag

④timer

e.g. timer handler function

import simplegui

def tick():

print “tick!”

timer = simplegui.create\_timer(1000,tick)

括号内第一个参数：毫秒数(1000ms=1s)

第二个参数：需要调用的函数或处理程序

timer.start() 启动timer

每一秒输出一个tick!（永不停止）

event queue:

不能同时处理两个事件

系统内部有事件队列，做完一个才会做另一个

(8) local v.s. global variables

在函数外部定义的变量都是global variables

在函数内部定义的变量是local variable

在函数内可以使用global variable，但是函数外不能使用local variable

在函数内修改global variable，在函数外不会对值有影响，除非声明：

e.g.

num = 4

def fun1():

global num

num = 5

def fun2():

global num

num = 6

print num 4

fun1()

print num 5

fun2()

print num 6

通常而言总要优先使用局部变量

but global variables are an easy way for event handlers- to communicate game information

e.g.

a = 3

b = 6

def f(a):

c = a + b

return c

global scope: a, b, f

local scope: a, c

(9) SimpleGUI

运行程序，出现一个frame（新的窗口）

左上角有control area控制区，可以返回值或者点击按钮之类的

右边是canvass画布，呈现信息

左下角时status area状态区，提供发生在画布上的键盘和鼠标事件的反馈

具体可以查看docs中的simple GUI专栏

先import simplegui

Create a frame: simplegui.create\_frame

set canvas background color: frame.set\_canvas\_background

start interactivity: frame.start

control objects: frame.add\_label, frame.add\_button 等

drawing on canvass: frame.set.draw\_handler

create a timer: simplegui.create\_timer

start a timer: timer.start

stop a timer: timer.stop

e.g. create red canvas:

import simplegui

frame = simplegui.create\_frame("My Frame", 100, 100)

frame.set\_canvas\_background("#FF0000")

frame.start()

或者

import simplegui

frame = simplegui.create\_frame("My Frame", 100, 100)

frame.set\_canvas\_background("red")

frame.start()

Recommended Program Structure:

①define globals (state)

②define helper functions 辅助函数

③define classes 类

④define event handlers 事件处理器

⑤create a frame

⑥register event handlers

⑦start frame & timers

e.g. counter with timer and click

import simplegui

counter = 0 ①

def increment(): ②

global counter

counter = counter +1

def tick(): ④

increment()

print counter

def buttonpress(): ④

global counter

counter = 0

frame = simplegui.create\_frame (“simpleGUI Test”, 100, 100) ⑤

timer = simplegui.create\_timer(1000,tick) ⑥

frame.add\_button(“Click me”,button press) ⑥

frame.start() ⑦

timer.start() ⑦

e.g. Basic Calculator

http://www.codeskulptor.org/#examples-buttons.py

Data: Store（第一个输入的数），Operand（第二个输入的数，操作数）

Operations: Print, Swap, Add, Subtract, Multiple, Divide

Store = store operation operand

import simplegui

store = 0

operand = 0

def output():

print “Store = ”, store

print “Operand = ”, operand

print “”

def swap ():

global store, operand

store, operand = operand, store

output()

def add():

global store

store = store + operand

output()

def sub():

global store

store = store – operand

output()

def mult():

global store

store = store \* operand

output()

def div():

global store

store = store / operand

output()

def enter(input):

global operand

operand = float(input) input是输入框输入的string

output()

f = simplegui.create\_frame (“Calculator”, 200, 200) 添加窗口（名字，宽，长）

f.add\_button(“Print”, output, 100) 添加按钮（名字，函数，宽度）

f.add\_button(“Swap”, swap, 100)

f.add\_button(“Add”, add, 100)

f.add\_button(“Sub”, sub, 100)

f.add\_button(“Mult”, mult, 100)

f.add\_button(“Div”, div, 100)

f.add\_input(“Enter operand”, enter, 100) 添加输入框（名字，函数，宽度）

frame.start()

http://www.codeskulptor.org/viz/#examples-input\_fields.py

注意：输入框输入text后，按回车，文本event handler for an input field get called

(10) Canvas and drawing

event-driven drawing

computer monitor- 2D grid pixels stored in frame buffer

computers update the monitor based on the frame buffer at the rate of around 60-72 times a second- refresh rate

many applications will register a special function called a “draw handler”

In CodeSkulptor, register the draw handler using a simpleGUI command.

CodeSkulptor calls the draw handler at around 60 times per second.

Draw handler updates the canvas using a collection of draw commands that include draw\_text, draw\_line, etc.

canvas coordinates:

simplegui.create\_frame(“Title”, width, height)

width, height: canvas的宽和高

origin [0,0]在左上角higher-left

后画的图案会覆盖先画的

define draw handler:

def draw(canvas):

canvas.draw\_text(“Hello”, [100,100], 24, “Whilte”)Text, position, font, color

绘制坐标是绘制图案的左下角lower-left

register event handlers:

frame = simplegui.create\_frame (“Testing”, 100, 100)

frame.set\_draw\_handler(draw)

frame.start()

exercise 5 ball radius:

import simplegui

# Define globals - Constants are capitalized in Python

HEIGHT = 400

WIDTH = 400

RADIUS\_INCREMENT = 5

ball\_radius = 20

# Draw handler

def draw(canvas):

canvas.draw\_circle([WIDTH/2, HEIGHT/2], ball\_radius, 1, "White", "White")

# Event handlers for buttons

def increase\_radius():

global ball\_radius

ball\_radius = ball\_radius + RADIUS\_INCREMENT

def decrease\_radius():

global ball\_radius

if ball\_radius > 5:

ball\_radius = ball\_radius - RADIUS\_INCREMENT

else:

ball\_radius = ball\_radius

# Create frame and assign callbacks to event handlers

frame = simplegui.create\_frame("Ball control", WIDTH, HEIGHT)

frame.set\_draw\_handler(draw)

frame.add\_button("Increase radius", increase\_radius)

frame.add\_button("Decrease radius", decrease\_radius)

# Start the frame animation

frame.start()

(11) String processing

print a, b a和b之间有空格，同一行

s = a + b a和b之间没有空格，同一行

print c[0] 取出字符串c的第一个字符**（从0开始计数）**

print c[-1] 取出字符串c的最后一个字符

print len(c) 获得c字符串的长度

e.g. 如果print len(c)的结果是14，那么print c[14]就会出现IndexError

print s[0:7] 获得从第0到第7（但不包括）的字符串，也就是0-6

print s[6:] 取得从第6开始直到最后的字符串

print s[:13] 获得从第0到第13（但不包括）的字符串，也就是0-12

s = str(375) 把数字转换成字符串

i= int(s) 把字符串转换成整型

s.count(“i”) 返回字符串s中字符i的个数

Strings can contain escape sequences:

|  |  |
| --- | --- |
| \' | single quote character |
| \" | double quote character |
| \\ | backslash character |
| \b | backspace character |
| \f | formfeed character |
| \n | new line character (starts a new line) |
| \r | carriage return character |
| \t | horizontal tab character (moves to next tab position, which is every eight characters) |
| \v | vertical tab character |
| \x<var>NN</var> | ASCII character represented by a hexidecimal number NN |

a.lower() 返回字符串a中所有字母全小写的新字符串

a.upper() 返回字符串a种所有字母全大写的新字符串

a.split(sep) 将sep字符从a字符串去掉，并切割开，形成一个List

e.g. convert money to dollars and cents

# Handle single quantity

def convert\_units(val, name):

result = str(val) + " " + name

if val > 1:

result = result + "s"

return result

# convert xx.yy to xx dollars and yy cents

def convert(val):

# Split into dollars and cents

dollars = int(val)

cents = int(round(100 \* (val - dollars)))

# Convert to strings

dollars\_string = convert\_units(dollars, "dollar")

cents\_string = convert\_units(cents, "cent")

# return composite string

if dollars == 0 and cents == 0:

return "Broke!"

elif dollars == 0:

return cents\_string

elif cents == 0:

return dollars\_string

else:

return dollars\_string + " and " + cents\_string

(12) Timer

time module:

time.time() 返回目前的时间（从一个公认的Epoch时间到现在的seconds）

timer = simplegui.create\_timer (interval, timer\_handler)

interval: milliseconds

start a timer:

timer.start()

stop a timer:

timer.stop()

e.g. screen saver

# Import modules

import simplegui

import random

# Global state

message = "Python is Fun!"

position = [50, 50]

width = 500

height = 500

interval = 2000

# Handler for text box

def update(text):

global message

message = text

# Handler for timer

def tick():

x = random.randrange(0, width)

y = random.randrange(0, height)

position[0] = x

position[1] = y

# Handler to draw on canvas

def draw(canvas):

canvas.draw\_text(message, position, 36, "Red")

# Create a frame

frame = simplegui.create\_frame("Home", width, height)

# Register event handlers

text = frame.add\_input("Message:", update, 150)

frame.set\_draw\_handler(draw)

timer = simplegui.create\_timer(interval, tick)

# Start the frame animation

frame.start()

timer.start()

exercise 4 expanding circle:

import simplegui

WIDTH = 200

HEIGHT = 200

radius = 1

# Timer handler

def tick():

global radius

radius += 1

# Draw handler

def draw(canvas):

global radius

canvas.draw\_circle([WIDTH/2, HEIGHT/2], radius, 1, "White")

# Create frame and timer

frame = simplegui.create\_frame("increase", WIDTH, HEIGHT)

timer = simplegui.create\_timer(100, tick)

frame.set\_draw\_handler(draw)

# Start timer

frame.start()

timer.start()

exercise 5 reflex test

import simplegui

total\_ticks = 0

first\_click = True

# Timer handler

def tick():

global total\_ticks

total\_ticks += 10

# Button handler

def click():

global first\_click

global total\_ticks

if first\_click == True:

total\_ticks = 0

print "First Click"

first\_click = not first\_click

else:

print "Time elapsed between button presses is "+ str(total\_ticks)+" hundredth of second."

first\_click = not first\_click

# Create frame and timer

frame = simplegui.create\_frame("Counter with buttons", 200, 200)

frame.add\_button("Click me", click, 200)

timer = simplegui.create\_timer(10, tick)

# Start timer

frame.start()

timer.start()

(13) Lists

one of the sequence types

用方括号包住一系列数字/字符串/别的list e.g. [1, 2, 3], [“hello”, “good bye”]

可以把List赋值给一个变量 e.g. position = [4, 9]

e.g.

l = [1, ,3, 4]

print l [1, 3, 4]

l2 = [“milk”, “eggs”, “bread”, “butter”]

print l2 [“milk”, “eggs”, “bread”, “butter”]

l3 = {[3,4], [“a”, “b”, “c”], []} 含有三个元素（三个lists）的list

print l3 {[3,4], [“a”, “b”, “c”], []}

大部分对string可用的函数，对list也有用：

print lent(l3) 3

print l[0] 1

print l3[1] [“a”, “b”, “c”]

print 12[1:3] [“eggs”, “bread”]

update elements: (mutation)

l2[0] = “cheese”

print l2 [“cheese”, “eggs”, “bread”, “butter”]

a = [1, 2, 3]

b = [1, 2, 3]

c = a

d = list(a)

print a is b False（单纯元素一样，不会认定相同）

print a is c True （这时候哪怕改变c，a也会改变）

print a is d False（list()函数复制内部元素，同b）

（只有列表有这种特性，列表是mutable的）

在list里面新增元素：list.append()

suitcase = []

suitcase.append("sunglasses")

减少list里面的元素：list.pop()（会返回去掉的元素）

list.pop() 去掉最后一个元素

list.pop(4) 去掉index为4的元素

去掉某个元素：list.remove()

list.remove(“yes”) 去掉元素yes（如果找不到会error）

in: check if sth. is in the list

e.g.

lst = [1, 82, -6, 4, 3, 8]

print 82 in lst True

index: give where sth. is in the list

e.g.

lst.index(8) 5

iteration over lists:

for element in list:

XXX

不要在遍历一个list的时候删掉元素，而是把要删掉的元素放在一个别的list里面，然后遍历那个要删除元素的List，在旧list当中找出它们来删掉

e.g.

remove = []

for num in numbers:

if num % 2 ==1:

remove.append(num)

for num in remove:

numbers.remove(num)

e.g.

一旦找到奇数就停止循环：

def check\_odd(numbers):

for num in numbers:

if num % 2 ==1:

return True return结束了函数

return False

e.g. square even number in list

def square\_list(numbers):

return [n\*\*2 for n in numbers if n % 2 == 0]

extend:

list\_extand([1, 2, 3]) 把1,2,3加入List

(14) tuple 多元组

another kind of sequence

but not mutable

用圆括号而非方括号 e.g. (4,5,6)

print type([4, 5, 6]) <class ‘list’>

print type((4, 5, 6)) <class ‘tuple’>

a = [4, 5, 6]

a[1] = 100

print a [4, 100, 6]

b = (4, 5, 6)

b[1] = 100

print b TypeError: ‘tuple’ does not support item assignment

(15) Keyboard input

e.g. 显示键盘所按的键

import simplegui

current\_key = “ ”

def keydown(key):

global current\_key

current key = chr(key) 数字编码→字符

def keyup(key):

global current\_key

current key = “ ”

def draw(canvas):

canvas.draw\_text(current\_key, [10, 25], 20, “Red”)

frame = simplegui.create\_frame (“Echo”, 35, 35)

frame.set\_keydown\_handler(keydown)

frame.set\_keyup\_handler(keyup)

frame.set\_draw\_handler(draw)

frame.start()

按下shift, ctrl这些的时候，chr函数返回的是类似于<16>，<17>的字符

e.g. 上下左右控制小球

WIDTH = 600

HEIGHT = 400

BALL\_RADIUS = 20

ball\_pos = [WIDTH/2, HEIGHT/2]

def draw(canvas):

canvas.draw\_circle[ball\_pos, BALL\_RADIUS, 2, “Red”, “White”]

def keydown(key):

vel = 4

if key == simplegui.KEY\_MAP[“left”]:

ball\_pos[0] -= vel

elif key == simplegui.KEY\_MAP[“right”]:

ball\_pos[0] += vel

elif key == simplegui.KEY\_MAP[“down”]:

ball\_pos[1] += vel

elif key == simplegui.KEY\_MAP[“up”]:

ball\_pos[1] -= vel

frame = simplegui.create\_frame (“Positional ball control”, WIDTH, HEIGHT)

frame.set\_draw\_handler(draw)

frame.set\_keydown\_handler(keydown)

frame.start()

exercise 4 ball radius control

import simplegui

WIDTH = 300

HEIGHT = 200

ball\_radius = 10

ball\_growth = 0

BALL\_GROWTH\_INC = .2

# Handlers for keydown and keyup

def keydown(key):

global ball\_growth

if key == simplegui.KEY\_MAP["up"]:

ball\_growth = BALL\_GROWTH\_INC

def keyup(key):

global ball\_growth

if key == simplegui.KEY\_MAP["up"]:

ball\_growth = 0

# Handler to draw on canvas

def draw(canvas):

global ball\_radius

ball\_radius += ball\_growth

# note that CodeSkulptor throws an error if radius is not positive

canvas.draw\_circle([WIDTH / 2, HEIGHT / 2], ball\_radius, 1, "White", "White")

# Create a frame and assign callbacks to event handlers

frame = simplegui.create\_frame("Home", 300, 200)

frame.set\_keydown\_handler(keydown)

frame.set\_keyup\_handler(keyup)

frame.set\_draw\_handler(draw)

# Start the frame animation

frame.start()

(16) motion

e.g. 自动运动的小球

import simplegui

WIDTH = 600

HEIGHT = 400

BALL\_RADIUS = 20

init\_pos = [WIDTH / 2, HEIGHT / 2]

vel = [5, 3] 定义速度为一个list（X,Y方向）

time = 0

def tick():

global time

time = time + 1

def draw(canvas):

ball\_pos = [0, 0] 定义位置为一个list（X, Y坐标）

ball\_pos[0] = init\_pos[0] + time \* vel[0] X方向坐标改变

ball\_pos[1] = init\_pos[1] + time \* vel[1] Y方向坐标改变

canvas.draw\_circle(ball\_pos, BALL\_RADIUS, 2, “Red”, “White”)

frame = simplegui.create\_frame (“Motion”, WIDTH, HEIGHT)

frame.set\_draw\_handler(draw)

timer = simplegui.create\_timer(100, tick)

frame.start()

timer.start()

可以根据微积分的原理去掉timer，直接简化为：

（内置计时器处理绘制函数为每1/60秒一次）

import simplegui

WIDTH = 600

HEIGHT = 400

BALL\_RADIUS = 20

ball\_pos = [WIDTH / 2, HEIGHT / 2]

vel = [5, 3] 定义速度为一个list（X,Y方向）

def draw(canvas):

ball\_pos[0] += vel[0] X方向坐标改变

ball\_pos[1] += vel[1] Y方向坐标改变

canvas.draw\_circle(ball\_pos, BALL\_RADIUS, 2, “Red”, “White”)

frame = simplegui.create\_frame (“Motion”, WIDTH, HEIGHT)

frame.set\_draw\_handler(draw)

frame.start()

collisions:

motion update

p[0] = p[0] + a \* v[0]

p[1] = p[1] + a \* v[1]

collision of ball p with ball:

left wall- p[0] <= r

right wall- p[0] >= width – r

bottom wall- p[1] >= height – r

up wall- p[1] <= r

reflection:

e.g. collide the left wall and reflect

v[0] = -v[0]

v[1] = v[1]

(17) mouse input

register:

frame.set\_mouseclick\_handler(click)

handler:

def click(position):

position: a pair of screen coordinates (i.e. a tuple of 2 integers)

e.g. draw a circle when clicking

def click(pos):

global ball\_pos

ball\_pos = list(pos) pos是tuple，转换成List，不然Immutable

def draw(canvas):

canvas.draw(ball\_pos, radius, 1, “Black”, “Red”)

frame = simplegui.create\_frame(“Mouse selection”, WIDTH, HEIGHT)

frame.set\_canvas\_background(“White”)

frame.set\_mouseclick\_handler(click)

frame.set\_draw\_handler(draw)

e.g. draw polyline

import simplegui

import math

WIDTH = 200

HEIGHT = 300

pos = [0, 0]

new = []

count = 0

# define mouseclick handler

def click(pos):

global new, count

new.append(pos)

if count == 2:

count = 1

if count == 1:

count = 2

if count == 0:

count = 1

# button to clear canvas

def clear():

global new, count

new = []

count = 0

# define draw

def draw(canvas):

if count == 1:

canvas.draw\_point(new[0], "White")

if count == 2:

canvas.draw\_line(new[0], new[1], 10, "White")

# create frame and register handlers

frame = simplegui.create\_frame("Echo click", 300, 200)

frame.set\_mouseclick\_handler(click)

frame.set\_draw\_handler(draw)

frame.add\_button("Clear", clear)

# start frame

frame.start()

(18) dictionary

a kind of mapping 映射

key → values 单向映射

用花括号建立一个dictionary

字典没有顺序

key可以是int, float, Boolean, string, tuple，但不能是list, dictionary

mutable objects cannot be key of dictionary

e.g.

d = {1:2, 3:4, 6:727, 83:421}

d2 = {‘abc’ : 47}

d3 = {‘ab’ : [1, 2, 3]} value可以是任何python objects

print d[1] 2

d[1] = 37 对key 1重新赋值37

e.g. encode words

CIPHER = {}

LETTERS = “abcdefghijklmnopqrstuvwxyz”

def init():

letter\_list = list(LETTERS) 把string转换为list

random.shuffle(letter\_list)

for ch in LETTERS:

CIPHER[ch] = letter\_list.pop()

message = ‘abc’

def encode():

emsg = ‘’

for ch in message:

emsg += CIPHER[ch]

print message, ‘encodes to’, emsg

def decode():

dmsg = ‘’

for key in CIPHER: 或者for key, value in CIPHER.items()

if ch == CIPHER[key]: if ch == value

dmsg += key

print message, “decodes to”, dmsg

利用dictionary简化程序：

e.g.

原程序：

def keydown(key):

global paddle1\_vel, paddle2\_vel

if key == simplegui.KEY\_MAP["up"]:

paddle2\_vel -= 2

elif key == simplegui.KEY\_MAP["down"]:

paddle2\_vel += 2

elif key == simplegui.KEY\_MAP["w"]:

paddle1\_vel -= 2

elif key == simplegui.KEY\_MAP["s"]:

paddle1\_vel += 2

新程序：

inputs = {"up": [1, -2],

"down": [1, 2],

"w": [0, -2],

"s": [0, 2]}

def keydown(key):

for i in inputs:

if key == simplegui.KEY\_MAP[i]:

paddle\_vel[inputs[i][0]] += inputs[i][1]

(19) images

loading images:

im = simplegui.load\_image(URL)

loading images across the web takes time, the image may not appear immediately the first time you try to load it into CodeSkulptor

drawing images:

canvas.draw\_image(im, src\_center, src\_size, dst\_center, dst\_size)

src\_center, src\_size: which portion of the image you gonna draw

dst\_center, dst\_size: draw on what part of the canvas

这四个参数都是一对数值

e.g.

http://www.codeskulptor.org/#examples-images.py

# Demonstration of a magnifier on a map

import simplegui

# 1521x1818 pixel map of native American language

# source - Gutenberg project

image = simplegui.load\_image("http://commondatastorage.googleapis.com/codeskulptor-assets/gutenberg.jpg")

# Image dimensions

MAP\_WIDTH = 1521

MAP\_HEIGHT = 1818

# Scaling factor

SCALE = 3

# Canvas size

CAN\_WIDTH = MAP\_WIDTH // SCALE

CAN\_HEIGHT = MAP\_HEIGHT // SCALE

# Size of magnifier pane and initial center

MAG\_SIZE = 120

mag\_pos = [CAN\_WIDTH // 2, CAN\_HEIGHT // 2]

# Event handlers

# Move magnifier to clicked position

def click(pos):

global mag\_pos

mag\_pos = list(pos)

# Draw map and magnified region

def draw(canvas):

# Draw map

canvas.draw\_image(image,

[MAP\_WIDTH // 2, MAP\_HEIGHT // 2], [MAP\_WIDTH, MAP\_HEIGHT],

[CAN\_WIDTH // 2, CAN\_HEIGHT // 2], [CAN\_WIDTH, CAN\_HEIGHT])

# Draw magnifier

map\_center = [SCALE \* mag\_pos[0], SCALE \* mag\_pos[1]]

map\_rectangle = [MAG\_SIZE, MAG\_SIZE]

mag\_center = mag\_pos

mag\_rectangle = [MAG\_SIZE, MAG\_SIZE]

canvas.draw\_image(image, map\_center, map\_rectangle, mag\_center, mag\_rectangle)

# Create frame for scaled map

frame = simplegui.create\_frame("Map magnifier", CAN\_WIDTH, CAN\_HEIGHT)

# register even handlers

frame.set\_mouseclick\_handler(click)

frame.set\_draw\_handler(draw)

# Start frame

frame.start()

可以将图片存在drop box，生成外链

但是一定要把www替换为dl

(20) object-oriented programming

object contains data

we can decide data type

each data type has its method (e.g. list.append)

OOP: 围绕objects，给这些objects创建新的类型(create new type)

e.g. Character

class Character: 下面def的都是methods

def \_\_init\_\_(self, name, initial\_health): 初始化器

self.name = name field ‘name’ inside object ‘self’

self.health = initial\_health

self.inventory = [] initiate field ‘inventory’ to blank list

def \_\_str\_\_(self):

s = "Name: " + self.name

s += " Health: " + str(self.health)

s += " Inventory: " + str(self.inventory)

return s create a string and return it

def grab(self, item):

self.inventory.append(item)

def get\_health(self):

return self.health

def example(): using the defined method

me = Character("Bob", 20) create an object via class\_obj = ClassName(..)

print me print an object会直接使用\_\_str\_\_ methond

me.grab("pencil") call class methods by object\_name.method(...)

me.grab("paper")

print me

print "Health:", me.get\_health()

example()

e.g. ball physics

http://www.codeskulptor.org/#examples-oo-ball.py

e.g. particle simulator

http://www.codeskulptor.org/#examples-particle\_class.py

# Particle class example used to simulate diffusion of molecules

import simplegui

import random

# global constants

WIDTH = 600

HEIGHT = 400

PARTICLE\_RADIUS = 5

COLOR\_LIST = ["Red", "Green", "Blue", "White"]

DIRECTION\_LIST = [[1,0], [0, 1], [-1, 0], [0, -1]]

# definition of Particle class

class Particle:

# initializer for particles

def \_\_init\_\_(self, position, color):

self.position = position

self.color = color

# method that updates position of a particle

def move(self, offset):

self.position[0] += offset[0]

self.position[1] += offset[1]

# draw method for particles

def draw(self, canvas):

canvas.draw\_circle(self.position, PARTICLE\_RADIUS, 1, self.color, self.color)

# string method for particles

def \_\_str\_\_(self):

return "Particle with position = " + str(self.position) + " and color = " + self.color

# draw handler

def draw(canvas):

for p in particle\_list:

p.move(random.choice(DIRECTION\_LIST))

for p in particle\_list:

p.draw(canvas)

# create frame and register draw handler

frame = simplegui.create\_frame("Particle simulator", WIDTH, HEIGHT)

frame.set\_draw\_handler(draw)

# create a list of particles

particle\_list = []

for i in range(100):

p = Particle([WIDTH / 2, HEIGHT / 2], random.choice(COLOR\_LIST))

particle\_list.append(p)

# start frame

frame.start()

对于定义的type- Point

已经赋值p = Point(3, 6)

如果 q = p，那么q和p其实是同一个object的不同名字，会一起改变

(q是别名)

如果p = point (coordinates), q = point(coordinates)，也会出现这种情况

e.g. use another class in one class:

# definition of Person class

class Person:

def \_\_init\_\_(self, first, last, year):

self.first\_name = first

self.last\_name = last

self.birth\_year = year

def full\_name(self):

return self.first\_name + " " + self.last\_name

def age(self, current\_year):

return current\_year - self.birth\_year

def \_\_str\_\_(self):

return "The person's name is " + self.full\_name() + ". Their birth year is " + str(self.birth\_year)

# definition of Student class

class Student:

# the person parameter must be a Person object

def \_\_init\_\_(self, person, pwd):

self.person = person

self.password = pwd

self.projects = []

# use the full\_name method for Person

def get\_name(self):

return self.person.full\_name()

def check\_password(self, pwd):

return pwd == self.password

def get\_projects(self):

return list(self.projects)

def add\_project(self, project):

self.projects.append(project)

e.g. image class

import simplegui

class ImageInfo:

def \_\_init\_\_(self, center, size, radius = 0, lifespan = None, animated = False):

self.center = center

self.size = size

self.radius = radius

if lifespan:

self.lifespan = lifespan

else:

self.lifespan = float('inf')

self.animated = animated

def get\_center(self):

return self.center

def get\_size(self):

return self.size

def get\_radius(self):

return self.radius

def get\_lifespan(self):

return self.lifespan

def get\_animated(self):

return self.animated

(21) while loop

e.g.

def countdown(n):

i = n

while i >= 0:

print i

i -= 1

countdown(5)

如果出现无限循环，会出现timelimit error

(22) sounds

e.g.

# simple music player, uses buttons and sounds

# note that .ogg sounds are not supported in Safari

import simplegui

# define callbacks

def play():

"""play some music, starts at last paused spot"""

music.play()

def pause():

"""pause the music"""

music.pause()

def rewind():

"""rewind the music to the beginning """

music.rewind()

def laugh():

"""play an evil laugh

will overlap since it is separate sound object"""

laugh.play()

def vol\_down():

"""turn the current volume down"""

global vol

if vol > 0:

vol = vol - 1

music.set\_volume(vol / 10.0)

volume\_button.set\_text("Volume = " + str(vol))

def vol\_up():

"""turn the current volume up"""

global vol

if vol < 10:

vol = vol + 1

music.set\_volume(vol / 10.0)

volume\_button.set\_text("Volume = " + str(vol))

frame = simplegui.create\_frame("Music demo", 250, 250, 100)

frame.add\_button("play", play,100)

frame.add\_button("pause", pause,100)

frame.add\_button("rewind",rewind,100)

frame.add\_button("laugh",laugh,100)

frame.add\_button("Vol down", vol\_down,100)

frame.add\_button("Vol up", vol\_up,100)

vol = 7

volume\_button = frame.add\_label("Volume = " + str(vol))

# load some sounds

music = simplegui.load\_sound("http://commondatastorage.googleapis.com/codeskulptor-assets/Epoq-Lepidoptera.ogg")

laugh = simplegui.load\_sound("http://commondatastorage.googleapis.com/codeskulptor-assets/Evillaugh.ogg")

# make the laugh quieter so my ears don't bleed

laugh.set\_volume(.1)

frame.start()

sounds is not associated with the frame, so even if the frame is closed, the sound will go on playing, until hitting the reset button.

(23) sprite

sprite: 2D image or animation integrated into a larger scene, usually treated as graphical overlay

在游戏中组织结构的一些逻辑元祖

sprite sheet: collection of sprites

set color using RGB model:

HTML string “rgb(255, 0, 0)” is equivalent to “Red”

add transparency:

add alpha channel to RGB model

HTML string “rgba(255, 0, 0, 0.5)” – 1 is opaque, 0 is transparent

sprite class

e.g. http://www.codeskulptor.org/#examples-sprite\_example.py

显示animation效果：

遍历tiled图片的各个部分即可——利用视觉残留persistence of vision

e.g. http://www.codeskulptor.org/#examples-asteroid\_animation.py

(24) sets

keep track of collection of data

list: ordered sequence of data

dictionary: key → value mapping

set: unordered collection of data with no duplicates

e.g.

s = set([1, 2, 3])

print s set([1, 2, 3])

s2 = set([1, 2, 2, 3])

print s2 set([1, 2, 3])

print s == s2 True

s2.add(4)

print s2 set([1, 2, 3, 4])

s2.remove(3)

print s2 set([1, 2, 4])

print s.intersection(s2) set([1, 2]) 返回重复部分，但是不改变s/s2

print 3 in s2 False

print 1 in s2 True

s.intersection\_update(s2)

print s set([1, 2]) 将重复部分赋值给s

s3 = set([1, 2, 3, 4, 5]).union(set([5, 6, 7]))

print s3 set([1, 2, 3, 4, 5, 6, 7])

s.update(set([2, 3])

print s set([1, 2, 3])

其他：

a\_set.difference(an\_iter)

return a set with all elements from set a\_set that are not iterable an\_iter

s\_set.difference\_update(an\_iter)

mutate a\_set to be the set difference of set a\_set and the set of elements in iterable an\_iter

a\_set.symmetric\_difference\_update(an\_iter)

mutate a\_set to be a set with all elements that are in exactly one of set a\_set and iterable an\_iter

a\_set.pop()

removes and returns an arbitrary element from set a\_set

a\_set.issubset(an\_iter)

returns whether the set a\_set is a subset of the set of elements in iterable an\_iter

s\_set.issuperset(an\_iter)

Returns whether the set a\_set is a superset of the set of elements in iterable an\_iter.

set是无序的，因此计算起来也更快

set也是mutable

在iteration的时候去掉某些元素，依旧最好先将元素储存在一个remove\_set里面，遍历之后一次性删除

或者：

for s in list(myset):

myset.remove(s)

因为list(myset)其实是对myset建立了一个副本